REMARKS

Claims 1-32 are pending in the application. Claims 1-8, 16-23, 31 and 32 stand rejected under 35 U.S.C. § 103(a) as being anticipated by Cabrera. In view of the following remarks, reconsideration and withdrawal of these grounds of rejection is requested.

Claim Rejections Under 35 U.S.C. § 103

Claims 1-8, 16-23, 31 and 32 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Cabrera et al. (U.S. Pat. No. 5,978,815). In view of the following remarks, reconsideration and withdrawal of this ground of rejection is respectfully requested.

The present invention comprises, in one exemplary embodiment, a Microsoft Windows® 9x computer operating system including 'user' mode code 10 and 'kernel' mode code 30 (See Fig. 1). The user mode code 10 comprises a system virtual machine 20 running software applications 21 and 22, and at least one MS-DOS virtual machine 25. The kernel mode code 30 comprises a virtual machine manager 40, a file system monitor 50, and an installable file system manager (IFSMGR) 60. The kernel mode code 30 also comprises system drivers 70-72, and a block Input/Output (I/O) subsystem 80. The block I/O subsystem 80 includes an I/O supervisor 81, a monolithic driver 82, and a layered plurality of device drivers 83-84.

In another exemplary embodiment, a Microsoft Windows NT® operating system is described which includes user mode code and kernel mode code. The user mode code includes at least one software application 100. The kernel mode code includes system services 110, I/O manager 120, and a file system monitor 50 for monitoring device drivers 130-132.

In either of the above-described exemplary embodiments, the file system monitor 50 filters 'file system' requests (i.e., I/O requests directed to a file system) by determining certain parameters each time a file system request is made. For example, the file system monitor 50 may determine whether the file system request is the first such request, or a subsequent request (See Fig. 3, Step 220).

If the file system request is the first such request, the file system monitor 50 is presumed to be the uppermost device driver in a 'device driver stack' (including the various device drivers referenced above with respect to the 9x and NT embodiments), and a calling module address of

the module making the file system request is stored for future reference (See Fig. 3, Steps 230, 231). The file system monitor 50 then filters the incoming request (Step 232).

If the file system request is a subsequent request, a calling module address for the I/O request must be determined (Step 221) and compared to the calling module address stored during the 'first' request (Step 222). If the calling module addresses are the same, file system monitor 50 continues to filter requests (Step 232). However, if the calling module addresses are different, the file system monitor 50 stops filtering, and denies future requests. The file system monitor 50 may be reinitiated, and filtering resume, through a re-hook process (See Fig. 4).

Claim 1 recites:

> A method for providing data security in a first device driver operably installed in a computer operating system having a layered plurality of device drivers for accessing data in a data storage device, the method comprising the steps of: detecting an I/O request to said first device driver; determining whether said first device driver is functionally uppermost in the layered plurality of device drivers; if said first device driver is functionally uppermost in the layered plurality of device drivers, performing the I/O request in said first device driver; and if said first device driver is not functionally uppermost in the layered plurality of device drivers, denying the I/O request in said first device driver, and allowing the I/O request to be performed by a next lower-level device driver in the layered plurality of device drivers. [emphasis added].

Thus, claim 1 requires a method for providing data security including the steps of "determining whether a first device driver is functionally uppermost in [a] layered plurality of device drivers" and "denying [an] I/O request" if the first device driver is not functionally uppermost in the layered plurality of device drivers. As explained below, Cabrera fails to disclose, teach or suggest such an invention.

Cabrera teaches a Microsoft Windows NT® operating system including a user mode and a kernel mode (see, col. 6, lines 60-62). The user mode includes at least one client process 94 which makes I/O requests 98 upon an I/O manager 110 in the kernel mode. The I/O manager 110 forwards each I/O request received from the client process 96 to the layer 1 driver 100 (see, col. 16, lines 34-39). The I/O request is forwarded through various additional 'layers' (e.g., layer 2 driver, layer 3 driver,...layer N driver) with each driver performing any required processing before forwarding the I/O request to the next driver (see, col. 16, lines 54-61).

Cabrera fails to disclose, teach or suggest a method for providing data security where the functional position of a "first device driver" is determined, and an "I/O request" is denied if the first device driver is not functionally uppermost in a layered plurality of device drivers, as recited in claim 1.

The Examiner asserts in the Office Action that Cabrera teaches a "first device driver" but does not specify which element corresponds to such claim language. With respect to the rejection of claim 1, the Examiner initially references the description relating to Figure 1 of Cabrera, which shows an apparent prior art operating system including a file system driver 26 and a device driver 28. Assuming, arguendo, the Examiner believes that the file system driver 26 corresponds to the "first device driver" of claim 1, Cabrera nowhere teaches or suggests a step of "determining whether [the] first device driver is functionally uppermost in [a] layered plurality of device drivers" (forming a device driver stack). In fact, Cabrera does not even teach a "layered plurality of device drivers" with reference to Figure 1; he teaches one additional driver at best (e.g., device driver 28). In sum, Cabrera does not disclose, teach or suggest determining the 'position' of the file system driver 26 (with respect to a device driver stack) in any way.

Further, Cabrera nowhere discloses or suggests "denying [an] I/O request" if the first device driver (e.g., file system driver 26) is not functionally uppermost in a "layered plurality of device drivers." Cabrera never discusses denial of I/O requests; Cabrera only discusses the transmission of I/O requests from the file system driver 26 to the device driver 28 via an I/O manager 40.

The Examiner additionally references Figure 5 of Cabrera in the rejection of claim 1, which arguably shows a layered device driver structure including device drivers 100, 102 and 104. Again, assuming for the sake of argument that the Examiner believes that the layer 1 device driver 100 corresponds to the "first device driver" of claim 1, Cabrera nowhere teaches or suggests a step of "determining whether [the] first device driver is functionally uppermost in [a] layered plurality of device drivers" or "denying [an] I/O request" if the first device driver (e.g., layer 1 driver 100) is not functionally uppermost in a "layered plurality of device drivers." As noted above, Cabrera never discusses denial of I/O requests. Accordingly, for all the reasons

discussed above, reconsideration and withdrawal of this ground of rejection with respect to claims 1-8 is respectfully requested.

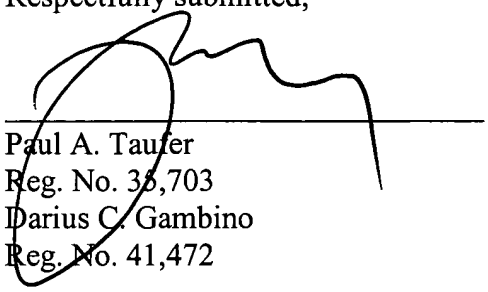
With specific reference to claim 6, Cabrera also fails to disclose, teach or suggest a method of determining whether a first device driver is functionally uppermost in the layered plurality of device drivers including steps of "determining whether said first device driver has been previously called," "detecting an initial calling module address," "storing [the] initial calling module address," "detecting a second calling module address," "comparing said second calling module address to the initial calling module address." As discussed above with reference to claim 1, Cabrera does not disclose, teach or suggest determining the relative position of the "first device driver" in a device driver stack. Thus, Cabrera cannot disclose the additional particulars of the position determining process recited in claim 6. Therefore, for this additional reason, reconsideration and withdrawal of this ground of rejection with respect to claim 6 is respectfully requested.

Independent claims 16, 31 and 32 all recite similar limitations to those discussed above with respect to independent claim 1. In particular, claim 16 describes a system which "determines whether [a] first device driver is functionally uppermost in the layered plurality of device drivers" and "denies [an] I/O request" if the "first device driver is not functionally uppermost in the layered plurality of device drivers." Claim 31 describes "computer-implemented instructions for determining whether [a] first device driver is functionally uppermost in the layered plurality of device drivers" and "computer-implemented instructions for denying the I/O request in said first device driver if said first device driver is not functionally uppermost in the layered plurality of device drivers." Claim 32 recites "means for determining whether [a] first device driver is functionally uppermost in the layered plurality of device drivers" and "means for denying the I/O request in said first device driver" if the "first device driver is not functionally uppermost in the layered plurality of device drivers." Hence, for at least those reasons discussed above with respect to independent claim 1, reconsideration and withdrawal of this ground of rejection with respect to claims 16-23, 31 and 32 is respectfully requested.

Conclusion

In view of the foregoing remarks, Applicants submit that this application is in condition for allowance at an early date, which action is earnestly solicited.

Respectfully submitted,



Paul A. Taufer
Reg. No. 38,703
Darius C. Gambino
Reg. No. 41,472

Piper Rudnick LLP
One Liberty Place
1650 Market Street, Suite 4900
Philadelphia, PA. 19103
Phone: 215.656.3300